Computational Science of Computer Systems

Méthodologies d'expérimentation pour l'informatique distribuée à large échelle

Martin Quinson

Université de Lorraine, Inria Nancy

Joint work with many colleagues: P. Bédaride, H. Casanova, P.N. Clauss, G. Corona, A. Degomme, F. Desprez, S. Genaud, A. Giersch, M. Guthmuller, Arnaud Legrand, Stephan Merz, L. Nussbaum, C. Rosa, M. Stillwell, Frédéric Suter, C. Thiéry, and many interns.

January 21th 2015
Rennes

About Me

Curriculum Vitæ

- 1999: Maîtrise (informatics) at Université de Saint Étienne
- 2003: PhD (informatics) at ENS-Lyon
- 2004: Post-Doctoral Researcher at University of California, Santa Barbara.
- 2004: Temporary teaching assistant at Université de Grenoble.
- Since 2005: Assistant professor at Université de Lorraine / Telecom Nancy
- 2011 – 2013: On leave at Inria Nancy – Grand Est
- March 2013: Habilitation Thesis
- 2013 – 2014: Leader of the Algorille joint team (Algorithms for the Grid)
- 2015 –: Member of the VeriDis joint team (Verification of Distributed Systems)
- Future? Professor in ENS-Rennes (team Myriads) ?

Research Common Theme since 15 years

- Discovery and Modeling of Large-scale HPC Systems (since my M.S work!)
- Make them usable by others: *e.g.*, provide performance models to schedulers
- One of the main contributors to SimGrid, a scientific instrument for such studies

Modern Computers are Large and Complex

Massive Parallelism

1.	Tianhe-2 (China)	3,120,000 cores	18MW
2.	Titan (USA)	560,640 cores	8MW
3.	Sequoia (USA)	1,572,864 cores	8MW
4.	K Computer (Japan)	705,024 cores	13MW
5.	Mira (USA)	786,432 cores	4MW

Computational Science $\rightsquigarrow$ ExaScale Systems

- Huge impact in all sciences and techniques and industries and businesses
- 1 Exaflop = 10^{18} operations. One million million million operations...

Not only in Computational Science

- Google dissipates 300MW ; Botnets control millions of zombie computers
- In addition, these systems are heterogeneous and dynamic

So, how do we *study* these beasts?

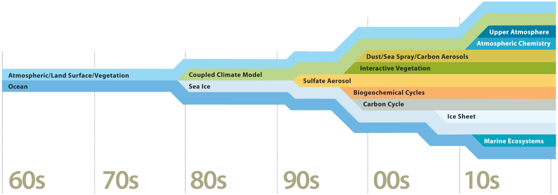
Computational Science of Computer Systems

My Research Field: Methodologies of Experimentation

- Goal: assess the performance and correctness of large-scale computer systems
- Question: Are we really producing scientifically sound results?
- Main contribution: SimGrid, a simulator of large-scale computer system

My approach: I am a physicist

- Empirically consider large-scale computer systems as natural objects
- Eminently artificial artifacts, but complexity reaches "natural" levels
- Other sciences routinely use computers to understand complex systems

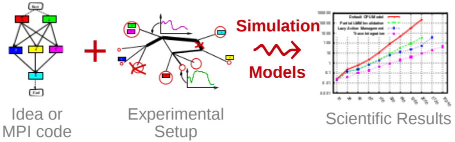


[PPL'09], cited 52 times.

Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

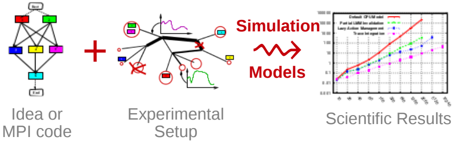
- Get preliminary results from partial implementations
- Experimental campaign with thousands of runs within the week
- Test your scientific idea, don't fiddle with technical subtleties (yet)



Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

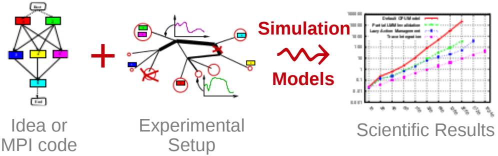
- Get preliminary results from partial implementations
- Experimental campaign with thousands of runs within the week
- Test your scientific idea, don't fiddle with technical subtleties (yet)

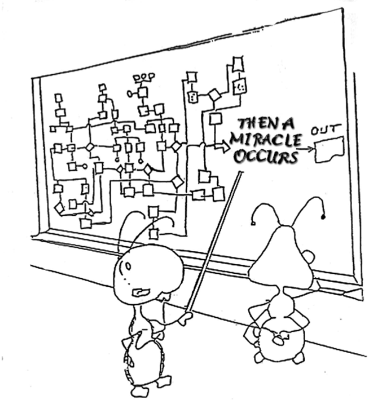


Simulation: Easiest Way to Study Distributed Applications

- Everything is actually centralized: Partially mock parts of your protocol
- No heisenbug: (Simulated) time does not change when you capture more data
- Clairevoyance: Observe every bits of your application and platform
- High Reproducibility: No or very few variability
- Capacity planning: *What if* network or CPU were faster
- Don't waste resources to debug and test

Simulation Challenges





Challenges for the Tool Makers

- Validity: Get realistic results (controlled experimental bias)
- Scalability: *Fast enough* and *Big enough*
- Open Science: Integrated lab notes, runner, post-processing (data provenance)

Simulation of Parallel/Distributed Systems

Network Protocols: Standards emerged: GTNetS, DaSSF, OmNet++, NS3
Huge amount of non-standard tools in other domains:

- Grid Computing — OptorSim | ChicagoSim | GridSim | JFreeSim | ...
- Peer-to-peer — P2Psim | SimP2P | PeerSim | OverSim | ...
- Volunteer Computing — SimBA | EmBOINC | SimBOINC | ...
- HPC/MPI — Dimemas | PSinS | BigSim | LogGoPSim | XSim | SST | ...
- Cloud Computing — CloudSim | GroudSim | iCanCloud | GreenCloud | ...

This raises severe methodological/reproducibility issues:

- Short-lived, badly supported (software QA), sparse validity assessment

Simulation of Parallel/Distributed Systems

Network Protocols: Standards emerged: GTNetS, DaSSF, OmNet++, NS3
Huge amount of non-standard tools in other domains:

- Grid Computing — OptorSim, ChicagoSim, GridSim, JFreeSim …
- Peer-to-peer — P2Psim, SimP2P, PeerSim, OverSim …
- Volunteer Computing — SimBA, EmBOINC, SimBOINC …
- HPC/MPI — Dimemas, PSinS, BigSim, LogGoPSim, XSim, SST …
- Cloud Computing — CloudSim, GroudSim, iCanCloud, GreenCloud …

This raises severe methodological/reproducibility issues:

- Short-lived, badly supported (software QA), sparse validity assessment

SimGrid: a 15 years old joint project

- Versatile: Grid, P2P, Clouds, HPC, Volunteer
- Collaborative: (ANR, CNRS, Univ., Inria) Open Source: active community
- Widely used: 150 publications by 120 individuals, 30 contributors

http://simgrid.org/ [UKSim'08] cited 350, [JPDC'14]

SimGrid Key Features: Fluid Network Model

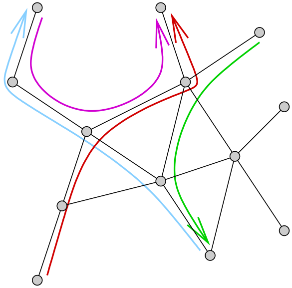
- Packet level models: Full net stack. Inherently slow, hard to instantiate
- Simple models: Delay-based, distribution, coordinates
  Very scalable, but no topology, *no network congestion*

SimGrid Key Features: Fluid Network Model

- Packet level models: Full net stack. Inherently slow, hard to instantiate
- Simple models: Delay-based, distribution, coordinates
  Very scalable, but no topology, *no network congestion*
- **Fluid models: Share bandwidth between flows** on macroscopic evts

Bandwidth sharing as an optimization problem

$$\sum_{\text{if flow i uses link j}} \rho_i \leqslant C_j$$



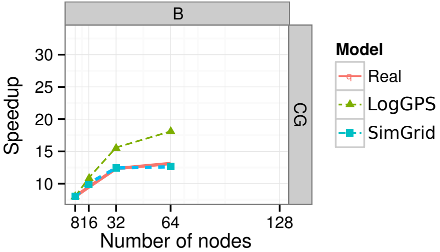
- Max-Min objective function: $\max(\min(\rho_i))$
- Reno fairness: $\max\left(\sum \arctan(\rho_i)\right)$
- Vegas fairness: $\max\left(\sum \log(\rho_i)\right)$

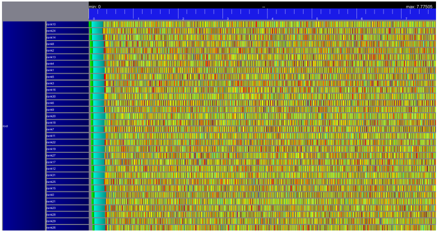
We implement, (in)validate and optimize these models since 10 years

- The classical "Observe, Analyze, Hypothesis, Test" loop

Validity Success Stories

unmodified NAS CG on a TCP/Ethernet cluster (Grid'5000)





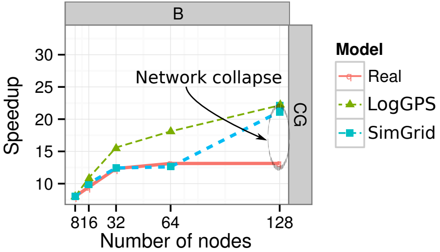
Key aspects to obtain this result

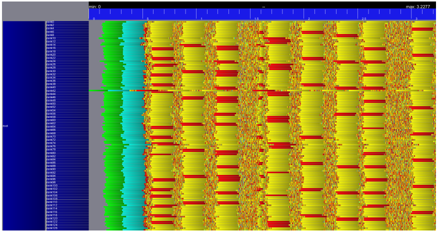
- Network Topology: Contention (large msg) and Synchronization (small msg)
- Applicative (collective) operations (stolen from real implementations)
- Instantiate Platform models (matching effects, not docs)
- All included in SimGrid but the instantiation (remains manual for now)

[IPDPS'11] cited 35, [SC'13]

Validity Success Stories

unmodified NAS CG on a TCP/Ethernet cluster (Grid'5000)





Discrepency between Simulation and Real Experiment. Why?

- Massive switch packet drops lead to 200ms timeouts in TCP!
- Tightly coupled: the whole application hangs until timeout
- Noise easy to model in the simulator, but useless for that very study
- Our prediction performance is more interesting to detect the real issue

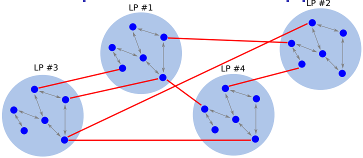
[IPDPS'11] cited 35, [SC'13]

Agenda

- Introduction
- Computational Science of Computer Systems (CS^2)
- Simulation Models
- Parallel Simulation of Discrete Event Systems
- Dynamic Verification of Distributed Applications
- Conclusion

Parallel Simulation of Discrete Event Systems

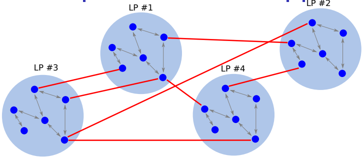
Classical Parallel Schema: split the whole applicative model



- Leads to good speedups (but still poor performance)
  dPeerSim: 4h → 1h when 2 LPs → 16 LPs (but 50s in sequential PeerSim)

Parallel Simulation of Discrete Event Systems

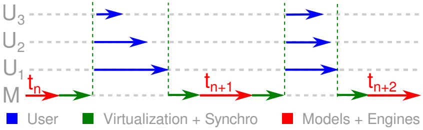
Classical Parallel Schema: split the whole applicative model

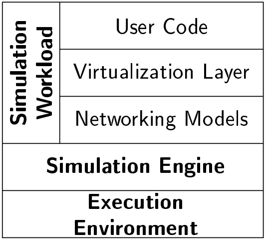


- Leads to good speedups (but still poor performance)
  dPeerSim: 4h → 1h when 2 LPs → 16 LPs (but 50s in sequential PeerSim)

New approach: Split at Virtualization layer (not in simulation engine)

- Virtualization contains threads (user's stack)
- Engine & Models remains sequential





- Synchronization costs of paramount importance

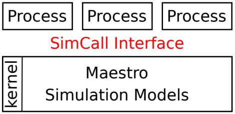
Efficient Parallel Fine-Grained Simulation

SimGrid is an Operating System

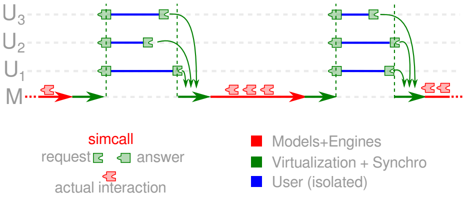
Simcalls separate processes, alleviating locking issues

- Very similar to syscalls in an operating system

Functional View



Temporal View



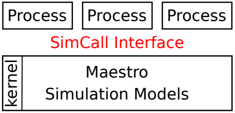
Efficient Parallel Fine-Grained Simulation

SimGrid is an Operating System

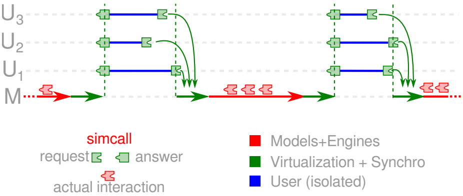
Simcalls separate processes, alleviating locking issues

- Very similar to syscalls in an operating system

Functional View

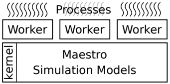


Temporal View

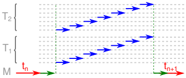


Leveraging Multicores

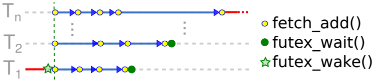
⇒ More processes than cores ⇝ Worker Threads (that execute co-routines ;)



Functional View



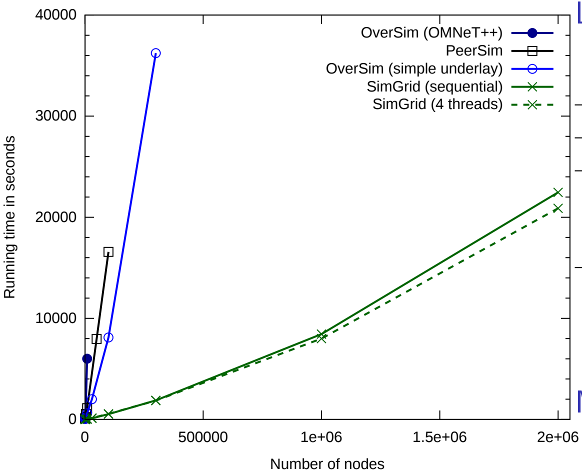
Temporal View



Ideal Algorithm

Performance Results

- Scenario: Initialize Chord, and simulate 1000 seconds of protocol
- Arbitrary Time Limit: 12 hours (kill simulation afterward)



Largest simulated scenario

	Size	Time
Omnet++	10k	1h40
PeerSim	100k	4h36
OverSim	300k	10h
SimGrid, seq	10k	32s
	300k	32mn
	2M	6h18
SimGrid//	10k	130s
	300k	40mn
	2M	5h55

Memory Usage

18kiB /process (stack: 12kiB)

First time that PDES is (a little) faster than DES [CCGrid'12], cited 17

Agenda

- Introduction
- Computational Science of Computer Systems (CS^2)
- Simulation Models
- Parallel Simulation of Discrete Event Systems
- Dynamic Verification of Distributed Applications
- Conclusion

Assessing the Correctness of HPC codes?

Writing Distributed Apps is notoriously difficult, but

The Good Old Days

- MPI codes circumvented issues with rigid communication patterns

- Performance First: fast code that rarely fail-stop $\ggg$ correct slow code

These Days are Now Over

- But rigid patterns do not scale! We now have to release the grip
- But this is dangerous! We now have to explicitly seek for correctness

Slowly, old ignored problems resurface

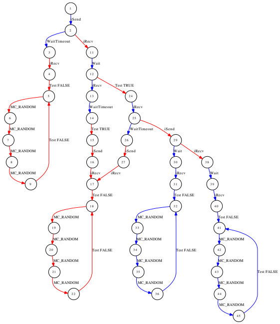
- When Tests are not enough anymore, turn to Formal Methods

Model Checking and Dynamic Verification

Automated Formal Methods

- Try to assess the correctness of a system by actively searching for faults
- If no fault found after an exhaustive search, correctness experimentally proved
- Dynamic Verification: Model Checking applied to real applications

Exhaustive Exploration

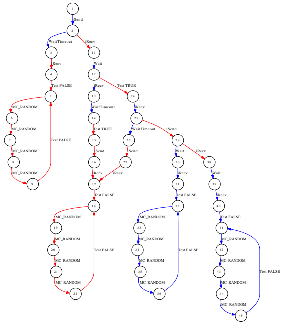


Model Checking and Dynamic Verification

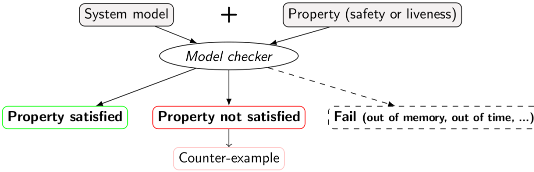
Automated Formal Methods

- Try to assess the correctness of a system by actively searching for faults
- If no fault found after an exhaustive search, correctness experimentally proved
- Dynamic Verification: Model Checking applied to real applications

Exhaustive Exploration



Model Checking: the Big Idea

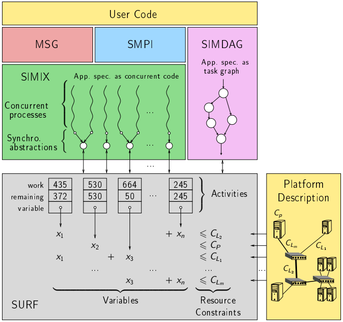


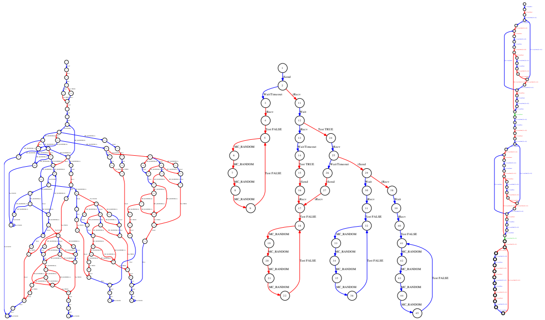
- My preferred outcome: a counter-example
- I tend to bug finding, not certification

SimGridMC: Formal Methods in SimGrid

Verify any application that would run in SimGrid

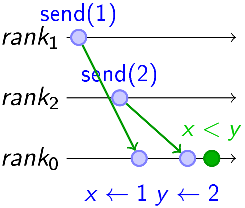
- Reuse the simulator's light virtualization to mediate apps' actions
- Replace the simulation kernel underneath with a model checker
- Tests all causally possible orders of events to dynamically verify the app
- System-level checkpoints the app to then rewind and explore another path
- This works with SMPI, and MSG (our simple API to study CSP algorithms)



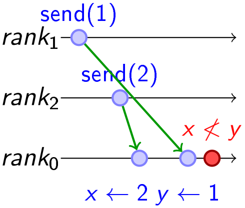


Example: Out of order receive

- Two processes send a message to a third one
- The receiver expects the message to be in order
- This may happen... or not




```
if (MPI_rank() == 0) {
  MPI_Recv(&x , MPI_ANY_SOURCE);
  MPI_Recv(&y , MPI_ANY_SOURCE);
  MC_assert(x < y);
} else {
  MPI_Send (&rank , 0);
}
```




```
*************************
*** PROPERTY NOT VALID ***
*************************
Counter-example execution trace:
[(1)recver] iRecv (dst=recver, buff=(verbose only), size=(verbose only))
[(3)sender] iSend (src=sender, buff=(verbose only), size=(verbose only))
[(1)recver] Wait (comm=(verbose only) [(3)sender -> (1)recver])
[(1)recver] iRecv (dst=recver, buff=(verbose only), size=(verbose only))
[(2)sender] iSend (src=sender, buff=(verbose only), size=(verbose only))
[(1)recver] Wait (comm=(verbose only) [(2)sender -> (1)recver])
```

Mitigating the State Space Explosion

Many execution paths are redundant $\rightsquigarrow$ cut exploration when possible

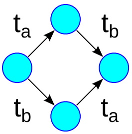
Dynamic Partial Ordering Reduction (DPOR)

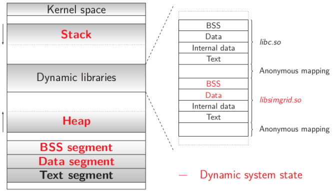
- Works on histories: test only one transitions' interleaving if independent
- *Independence theorems:* Local events are independent; iSend+iRecv also; ...
- Must be conservative (exploration soundness at risk!)
- It works well (for safety properties)

System-Level State Equality

- Works on states: detect when a given space was previously explored
- Complementary to DPOR (but not compatible yet)
- Introspect the C/C++/Fortran app just like gdb (+some black magic)

[AVOCS'10]





[PDP'15]

Some Results

Wild safety bug in our Chord implementation (≈ 500 lines of C)

- Simulation: bug on large instances only; MC finds small trace (1s with DPOR)

Mocked liveness bug

- Buggy centralized mutual exclusion: last client never obtains the CS
- About 100 lines – state snapshot size: 5Mib; Verified with up to 7 processes

Verifying MPICH3 complience tests

- Looking for assertion failures, deadlocks and non-progressive cycles
- 6 tests; ≈ 1300 LOCs (per test) – State snapshot size: ≈ 4MB
- We verified several MPI2 collectives too ☺ (but all good so far ☹)

Protocol-wide Properties

- e.g, Send-deteministic: On each node, send and recv evts always in same order (allows more efficient application checkpointing)
- Even harder than liveness properties (not LTL), but doable in SimGrid

Much more to say about SimGrid (too little time)

Hybrid Network Models

- Fluid model: model contention in steady state for large messages
- LogOP model: model intra-node delays and synchronization
- Also: MPI collectives, TCP (slow-start, cross-traffic), soon IB

Realistic Emulation

- SMPI: Study real MPI applications within SimGrid
- Simterpose: Study real arbitrary applications (ongoing)

High Performance Simulation

- Fast Enough: Innovative PDES; Efficient algorithms and implementations
- Big Enough: Scalable and versatile platform representation

Formal Verification of Distributed Apps

- Safety, Liveness or CTL properties, with DPOR or state equality

Research Project for the Next 10 Years

Make Large-Scale Distributed Systems Easier to Use

- They are pervasive in our connected societies, yet almost uncontrolled
- Research Plan: *Computational Science of Computer Systems*
  - Leveraging computers to understand computers
- Expected Visible Outcome: Propose a valgrind-like for Distributed Systems
- This is perfectly in line with what I'm doing since 15 years

Why in Myriads?

- Distributed Systems, with focus on experimentation (Grid'5000, etc)
- Many works that solve hard OS-level issues to help distributed systems

Why at ENS Rennes?

- We need more teachers that are proficient with Systems Internals
- Teaching of paramount importance to me. Lots of activities on teaching CS:
  - Unplugged activities; Programming exercisers; Research groups; National Days
- More on this on Friday 12:30 :)

What is SMPI?

- Reimplementation of MPI on top of SimGrid
- Imagine a VM running real MPI applications on platforms that does not exist
  - Horrible over-simplification, but you get the idea
- Computations run for real on your laptop, Communications are faked

What is it good for?

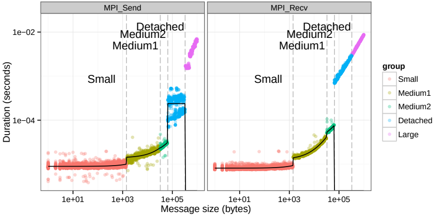
- Performance Prediction ("what-if?" scenarios)
  - Platform dimensioning; Apps' parameter tuning
- Teaching parallel programming and HPC
  - Reduced technical burden
  - No need for real hardware, or hack your hardware

Studies that you should NOT attempt with SMPI

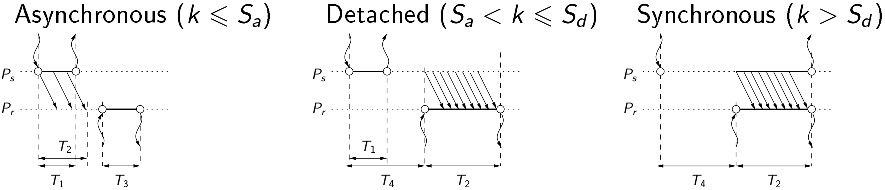
- Predict the impact of L2 caches' size on your code
- Interactions of TCP Reno vs. TCP Vegas vs. UDP
- Claiming a simulation of 1000 billions nodes

SimGrid Network Model

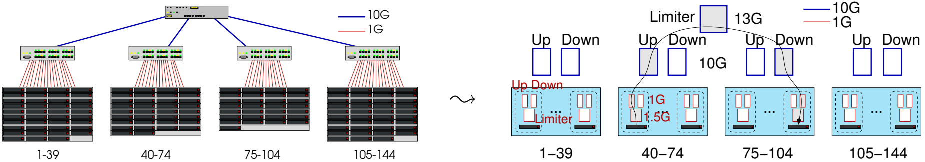
Measurements



Hybrid Model



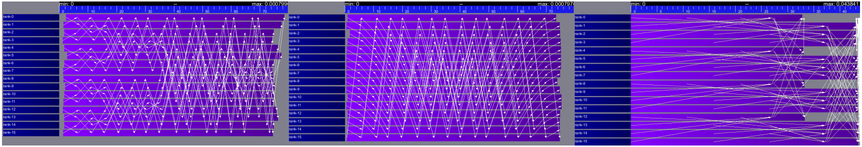
Fluid model: account for contention and network topology



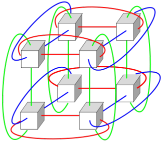
SimGrid Modeling of MPI

MPI Collectives

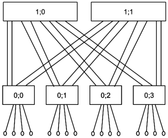
- SimGrid implements more than 120 algorithms for the 10 main MPI collectives
- Selection logic from OpenMPI, MPICH can be reproduced



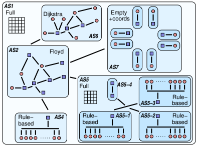
HPC Topologies



Torus



Fat-trees



Hierarchies of ASes

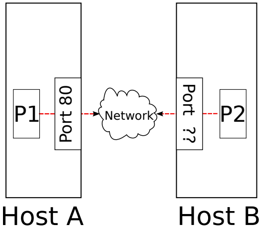
But also

- External load (availability changes), Host and link failures, Energy (DVFS)
- Virtual Machines, that can be migrated; Random platform generators

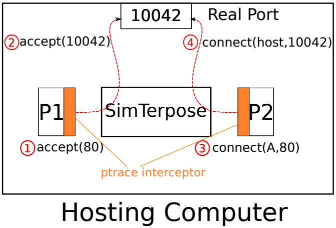
SimTerpose Project

Dream: Simulate any applications on top of SimGrid

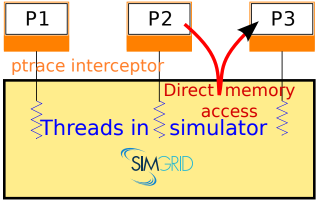
Simulated Setup



Take 1: ptrace plumbering



Take 2: Full Emulation



Current State

- Functional POC: send/recv exchange
- Need to handle the other 200 syscalls
  - Intercept, store metadata
  - Inform simulator, report effect on procs
- Time and DNS need love at link time
- We are redeveloping a libC! (in strange way ;)